(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－272623

(43)公開日　平成11年(1999)10月８日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	15/16	３８０	Ｇ０６Ｆ	15/16	３８０Ｚ
	9/46	３６０		9/46	３６０Ｃ
	13/00	３５７		13/00	３５７Ｚ

審査請求　未請求　請求項の数７　ＯＬ　（全 15 頁）

(21)出願番号　特願平10－70493

(22)出願日　平成10年(1998)３月19日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　熊本　乃親
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　田中　竜太
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　弁理士　今村　辰夫　（外１名）

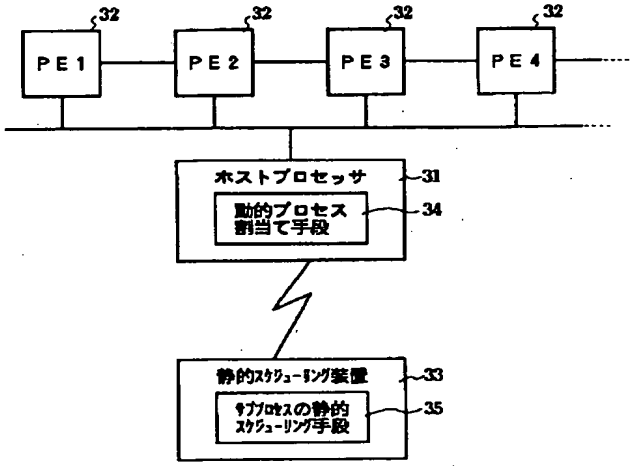
(54)【発明の名称】　プロセス割当て方法、静的スケジューリング装置、及び記録媒体

(57)【要約】

【課題】本発明は、プロセス割り当て方法、静的プロセス割り当て装置及び記録媒体に関し、リニアアレイ型プロセッサシステムにおける動的プロセス割当てによってシステム全体の能力を最大限引き出すために、サブプロセスの実行スケジュールを、予め静的スケジューリング手段によって求めておくことで、効率的な処理を可能にする。

【解決手段】複数のプロセッサ３２を直列接続したリニアアレイ型マルチプロセッサシステムの各プロセッサ３２に対し、例えば、ホストプロセッサ３１の動的プロセス割当て手段３４が複数のサブプロセスを構成要素として含むプロセスを動的に割当てる時、静的スケジューリング装置３３に設けたサブプロセスの静的スケジューリング手段３５が、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておくようにした。

システムの説明図



1

【特許請求の範囲】
【請求項１】複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるプロセス割当て方法において、
前記動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、サブプロセスの静的スケジューリング手段により静的に求めておくことを特徴としたプロセス割当て方法。
【請求項２】前記サブプロセスの静的スケジューリング手段が、そのプロセスが要求するプロセッサ数最小となるスケジュール情報を得ることを特徴とした請求項１記載のプロセス割当て方法。
【請求項３】前記サブプロセスの静的スケジューリング手段が、そのプロセス単体の処理時間最小となるスケジュール情報を得ることを特徴とした請求項１記載のプロセス割当て方法。
【請求項４】前記サブプロセスの静的スケジューリング手段が、そのプロセスが要求するプロセッサ間データ転送量最小となるスケジュール情報を得ることを特徴とした請求項１記載のプロセス割当て方法。
【請求項５】前記サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュール情報を選択して割当てることを特徴とした請求項１記載のプロセス割当て方法。
【請求項６】複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるためのサブプロセスのスケジュール情報を、予め、静的に求めておくサブプロセスの静的スケジューリング手段を備えていることを特徴とした静的スケジューリング装置。
【請求項７】コンピュータに、
複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるためのサブプロセスのスケジュール情報を、予め、静的に求めておく手順を実行させるためのプログラムを記録したコンピュータ読取可能な記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるプロセス割当て方法、そのための静的スケジューリング装置、及び記録媒体に関する。
【０００２】近年、ディジタル通信システム、マルチメディアシステムなど、実時間処理が要求される分野においては、高速、低コストで、かつ複雑なアルゴリズムに

2

柔軟に対応可能なディジタル信号処理プロセッサ（ＤＳＰ）を始めとして、信号処理を高速に実行可能なプロセッサが広く使われている。
【０００３】ところが、画像処理、音響処理ＣＧ、などの処理を複数同時に処理しようとした場合、単一のプロセッサでは実時間で処理を行うことが困難となっている。また、単一プロセッサで実現される処理においても、処理すべきデータ量が増加した場合や、要求される速度が早くなった場合、単一のプロセッサでは実現できなくなってしまうことがある。そこで、複数のプロセッサを使用して処理の高速化を図る必要がある。
【０００４】
【従来の技術】以下、図に基づいて従来例を説明する。
§１：従来例１の説明
従来、マルチプロセッサシステムにおいて、各プロセッサにプロセスを割当てる場合、空いているプロセッサから順番にプロセスを割当てる方法がとられている。これは、リニアアレイ型マルチプロセッサシステムではプロセッサ間通信を行う場合、転送経路が１つ（直線状）又は２つ（リング状）しかないため、通信時間をほとんど考慮する必要がないためである。
【０００５】また、このように空いているプロセッサを見つけて順に割当てる方法では、動的なプロセス割当てに際し、重要な割当てに要する時間が小さいという利点がある。しかしながら、アプリケーションによっては、プロセッサ間のデータ転送に用いるリンクに流れるデータ転送量がかなり多い場合があり、このような場合、或る特定のプロセッサ間のリンクのデータ転送量が大きくなり、他のプロセッサ間のリンクのデータ転送流が小さくなってしまうことが起こる。
【０００６】このような場合、プロセッサの能力はまだ十分に空いているにも関わらず、データ転送ができないため、全体の処理が実時間で実現不可能になってしまうことがある。
【０００７】§２：従来例２の説明
データ転送量を考慮した割当て方法としては、例えば、特開平９－３４８４７号公報に記載された発明がある。以下、この例を従来例２として説明する。従来例２では、通信路にスイッチを置き、そのスイッチでデータ転送量を計数し、それに応じて通信経路を変更することで、データ転送量を考慮したプロセスの割当てを行う。
【０００８】ところが、このプロセス割当て方法では、通信経路が複数あれば効果はあるが、リニアアレイ型プロセッサシステムでは、通信路は１つ、リンク構成でも２つしかなく、ほとんど効果がない。
【０００９】§３：従来例３の説明
プロセッサ間のデータ転送量の最大値が最小となるプロセスを割当てることで、プロセッサの能力を生かした処理の実現を可能にしたものが既に提案されている。ところが、ここで割当てる各プロセスは、一般に複数のプロ

セッサ資源を要求するため、単一プロセッサで処理されるサブプロセスを複数接続した構成になっている。

【００１０】従って、このサブプロセスの実行スケジュールによって、そのプロセスが要求する資源は変化する。このようなサブプロセスの実行スケジュールについては、特開平８－１５２９０３号公報に示されているように、予め制御グループを設定し、グループ内をスケジューリングする方法が知られている。

【００１１】しかし、この方法では、制御グループを先に静的に決定してしまうため、グループ間のデータ転送量の負荷分散を行うには、その都度、制御グループの割当て変更が必要となり、負荷分散が動的に行われない。

【００１２】§４：従来例４の説明

複数プロセッサで構成されるシステムの中で、複数のプロセッサを直列接続して得られるリニアアレイ型マルチプロセッサシステムが、先に提出した特願平９－２２１６１７号（以下「従来例４」と記す）に記載されている。この従来例４に示されているリニアアレイ型マルチプロセッサシステムは、構成が簡単で、最も安価なシステムを構成できる。

【００１３】前記リニアアレイ型マルチプロセッサシステムでは、プロセッサは左右のいずれかのプロセッサへは、直接データ転送できるが、それ以外のプロセッサへのデータ転送は、他のプロセッサを経由してしかデータ転送できない。従って、複数のプロセッサに対し、ユーザの要求によりダイナミックに処理を行ったり、停止する場合、プロセスの割当て方によって、複数のプロセッサの能力を十分に生かしきれない。

【００１４】以下、従来例４について、図を参照しながらその概要を説明する。図８は従来例の説明図（その１）であり、信号処理アクセラレータ（リニアアレイ型プロセッサシステム）の説明図である。以下、図８に基づいて、信号処理アクセラレータ（リニアアレイ型プロセッサシステム）の構成を説明する。

【００１５】図８に示した信号処理アクセラレータは、複数で同一の情報処理ユニット１０を含む。各情報処理ユニット１０は互いに接続されると共に、ホストメモリバス３０に接続される。各情報処理ユニット１０は、信号処理プロセッサ１１、命令キャッシュ１２、データＲＡＭ１３、リンク制御部１４、１５、メインキャッシュ１６、リンクキャッシュ１７、ＤＲＡＭコントローラ１９を含む。なお、前記信号処理プロセッサ１１及びデータＲＡＭ１３は、信号処理部２５を構成する。また、リンク制御部１４、１５、メインキャッシュ１６、及びリンクキャッシュ１７は、通信制御部２６を構成する。リンク制御部１４、１５には通信リンク２０が接続される。各情報処理ユニット１０は通信リンク２０を介して直列に配列され、隣接する情報処理ユニット１０と通信する。通信内容を或る情報処理ユニット１０から次の情報処理ユニット１０へと伝達していくことで、任意の情報処理ユニット間で通信を行うことができる。この場合、図８では、３個の情報処理ユニット１０が示されているが、個数は３個に限らず任意である。

【００１６】また、各情報処理ユニット１０はＤＲＡＭコントローラ１９を介してホストメモリバス３０に接続される。更に、ホストプロセッサ３１がホストメモリバス３０に接続される。信号処理プロセッサ１１は信号処理機能を実現するプロセッサであり、命令キャッシュ１２は信号処理プロセッサ１１が頻繁に用いる命令を格納しておくためのキャッシュメモリである。

【００１７】なお、信号処理プロセッサ１１が実行するプログラムは、命令キャッシュ１２以外にＤＲＡＭ１８に格納されている。データＲＡＭ１３は、信号処理プロセッサ１１がデータ処理する際に中間結果を格納するため等のワーク領域として用いられる。メインキャッシュ１６及びリンクキャッシュ１７は、信号処理プロセッサ１１が処理するデータを格納しておくためのキャッシュメモリである。

【００１８】メインキャッシュ１６には、情報処理ユニット１０自身のＤＲＡＭ１８から取り込んだデータを格納し、リンクキャッシュ１７には、リンク制御部１４、１５を介して他の情報処理ユニット１０から取り込んだデータを格納する。

【００１９】メインキャッシュ１６のデータがスワップアウトされても、そのデータが必要になった場合には、自分のＤＲＡＭ１８からデータを再度読み出すことができる。それに対して、リンクキャッシュ１７のデータがスワップアウトされると、別の情報処理ユニット１０から通信リンク２０を介してデータを再度持ってくる必要がある。

【００２０】メインキャッシュ１６とリンクキャッシュ１７を同一のキャッシュメモリとしてしまうと、例えば、通信負荷が重い状態なのに、自身のＤＲＡＭ１８からのデータをキャッシュメモリに格納することによって、他の情報処理ユニット１０から獲得したデータをスワップアウトしてしまう等の問題が生じる。この例では、メインキャッシュ１６とリンクキャッシュ１７とを機能別に分けている。

【００２１】データ処理を行う際、複数の情報処理ユニット１０は互いに通信しながら、並列処理、或いはパイプライン処理を行う。たとえば、いくつかの情報処理ユニット１０が画像データ処理を並列に実行している間に、別の幾つかの情報処理ユニット１０が音声データを並列に処理すること等が可能である。

【００２２】複数の情報処理ユニット１０間の通信は、通信リンク２０によって行われる。従って、ホストメモリバス３０は、複数の情報処理ユニット１０間での通信には全く関与せず、ホストプロセッサ３１が実行するＯＳプロセス等の他のプロセスにデータ転送経路を提供できる。

5

【００２３】各情報処理ユニット１０は処理後のデータをＤＲＡＭ１８に書き込む。ホストプロセッサ３１はホストメモリバス３０を介してＤＲＡＭ１８にアクセスすることによって、データ処理後のデータを書き込むことができる。図８の信号処理アクセラレータは、ホストメモリバス３０を介在しないで通信可能な複数の情報処理ユニット１０を設けて、それら情報処理ユニット１０に並列処理を実行させるので、バス競合によりデータ処理速度が低下することなく、高速な信号処理を実現することができる。

【００２４】また、画像処理プロセスや音声処理プロセス等の複数のプロセスの各々に、別の情報処理ユニット１０を割当てることができるので、複数の異なった信号を処理する必要があるマルチメディア信号処理に適している。また、信号処理部２５（信号処理プロセッサ１１、命令キャッシュ１２、データＲＡＭ１３）、通信制御部２６（キャッシュメモリ１６、及び１７と、リンク制御部１４及び１５）、及びメモリ（ＤＲＡＭ１８及びＤＲＡＭコントローラ１９）を１つのチップ上に集積回路化して、従来のメモリと同等な形でパーソナルコンピュータに搭載することができる。

【００２５】従って、コストを従来のメモリバスに重複させることが可能であると共に、メモリバス内に埋め込まれた信号処理アクセラレータをソフトウェアで活用できるという利点がある。従って、ハードウェアの追加、拡張にかかる費用を削減すると共に、機能拡張に優れたシステムを構築できる。

【００２６】§５：プロセス割当て方法の説明・・・図９参照

図９は従来例の説明図（その２）であり、２つの異なったプロセス割当ての方法（Ａ図は例１、Ｂ図は例２）を示した図である。以下、図９に基づいて、２つの異なったプロセス割当ての方法を説明する。図９のＡ図のように、プロセス１をプロセッサエレメントＰＥ１及びＰＥ３に割当て、プロセス２をプロセッサエレメントＰＥ２及びＰＥ４に割当てた場合を説明する。

【００２７】一つのプロセッサ内で２つのＰＥ間のデータ転送量をＭとすれば、ＰＥ１及びＰＥ３間でＰＥ２を介してのデータ転送量はＭとなり、また、ＰＥ２及びＰＥ４間でＰＥ３を介してのデータ転送量もＭとなる。従って、データ転送量は、ＰＥ１及びＰＥ２間でＭ、ＰＥ２及びＰＥ３間で２Ｍ、ＰＥ３及びＰＥ４間でＭとなる。

【００２８】次に、図９のＢ図に示したように、プロセス１をプロセッサエレメントＰＥ１及びＰＥ２に割当て、プロセス２をプロセッサエレメントＰＥ３及びＰＥ４に割当てた場合を説明する。この場合、ＰＥ１及びＰＥ２の間でのデータ転送量がＭ、ＰＥ３及びＰＥ４間でのデータ転送量がＭとなり、ＰＥ２及びＰＥ３間ではデータ転送は行われない。

6

【００２９】各ＰＥ間のデータ転送能力が、たとえば、１．５Ｍであるとすると、図９のＡ図の場合（例１）には、一方のプロセスの処理が実現不可能となるのに対して、図９のＢ図の場合（例２）には、両方のプロセスの処理を同時に実現できる。このようにプロセスの割当て方によっては各リンクのデータ転送量が異なる結果となり、複数のプロセスの完全な同時処理が可能な場合と、不可能な場合とが生じる。

【００３０】同時処理が不可能な場合には、当然ながら全体でのデータ処理速度が劣化することになる。しかも、他のプロセッサエレメントから要求される数やタイミングは、実際の要求が発行される前には全く未知であるため、プロセスの割当てを動的に行う必要がある。従って、効率の良い動的プロセス割当てアルゴリズムが必要になる。

【００３１】§６：動的プロセス割当てアルゴリズムの説明・・・図１０～１３参照

図１０～図１３は従来例の説明図（その３）～（その６）であり、動的プロセス割当てアルゴリズムを示した図である。以下、図１０～図１３に基づいて、動的プロセス割当てアルゴリズムを説明する。

【００３２】(1)：メインルーチンの説明・・・図１０参照

図１０は従来例の説明図（その３）であり、動的プロセス割当てアルゴリズムのメインルーチンを示すフローチャートである。以下、図１０に基づいて、動的プロセス割当てアルゴリズムのメインルーチンを説明する。なお、Ｓ１～Ｓ５は各処理ステップを示す。

【００３３】この動的プロセス割当てアルゴリズムは、リソース管理プログラム（図示省略）が実行する処理である。この動的プロセス割当てアルゴリズムは、以下の２つの指標に基づいてリソース（この場合のリソースはＰＥを指す）の割当てを行う。第１の指標は、割当てるプロセスのデータ転送ができるだけ他のデータ転送と重ならないことである。第２の指標は、当該プロセスを割当てた結果、次のプロセスを割当てる際に、できるだけ他のデータ転送と重ならないことである。

【００３４】先ず、第１の指標によって、割当てようとしているプロセスによって生じる通信リンク２０上のデータ転送量の最大値が最小になるように設定する。次に、同一の最大値を有する割当て方の中から、第２の指標を用いて、次に割当てるプロセスができるだけ阻害されないような割当て方を求める。

【００３５】図１０に示したように、このアルゴリズムにおいては、ＰＥを一個要求する場合と、複数個要求する場合とで、別々に割当て方を求める。これはＰＥを１個だけ要求する場合には、当該プロセスによる通信リンク２０上のデータ転送は生じないために、次のプロセス割当てに対する影響を考えればよいからである。

【００３６】それに対してＰＥを複数個割当てる場合に

は、通信リンク２０を介したデータ転送が必要であるため、そのプロセスの割当て方によって、当該プロセス自身の効率が影響を受けてしまう。

【００３７】図１０のステップＳ１において、フリーリソースとして利用可能なＰＥの数を検査する。利用可能なＰＥが存在しない場合にはアルゴリズムを終了し、存在する場合にはステップＳ２に進む。ステップＳ２において、要求されるＰＥの数が１個であるのか否かを判断する。１個の場合はステップＳ３に進み、複数の場合はステップＳ４に進む。

【００３８】ステップＳ３において、ＰＥの１個要求に対する割当てを行う。割当てが失敗した場合にはアルゴリズムを終了し、成功した場合は次にステップＳ５に進む。ステップＳ４において、ＰＥの複数個要求に対する割当てを行う。割当てが失敗した場合にはアルゴリズムを終了し、成功した場合にはステップＳ５に進む。ステップＳ５において、プロセスＩＤを更新する。すなわち、新規のプロセスに、新規のプロセスＩＤを割当てる。これでアルゴリズムを終了する。

【００３９】(2)：ＰＥの１個要求に対する割当ての説明・・・図１１参照

図１１は、従来例の説明図（その４）であり、図１０のステップＳ３におけるＰＥの１個要求に対する割当て処理を示すフローチャートである。なお、Ｓ１１～Ｓ１７は各処理ステップを示す。

【００４０】ステップＳ１１において、利用可能なＰＥを検索する。次に、ステップＳ１２において、全ての利用可能なＰＥに対してループを構成する。すなわち、全ての利用可能なＰＥに対して以下のステップを繰り返す。ステップＳ１３において、１個のＰＥを仮割当てする。ステップＳ１４において、次の割当て効率を計算する。なお、割当て効率を計算した結果を「Ｒｅｓｕｌｔ」と記す。

【００４１】ステップＳ１５において、Ｒｅｓｕｌｔが最小の値を保持する。ステップＳ１６においてループを終了する。ステップＳ１７において、Ｒｅｓｕｌｔが最小の値であるＰＥを、実際にプロセスに割当てる。以上で処理を終了する。

【００４２】(3)：ＰＥの複数個要求に対する割当ての説明・・・図１２参照

図１２は従来例の説明図（その５）であり、ＰＥの複数個要求に対する割当てを示すフローチャートである。以下、図１０のステップＳ４におけるＰＥの複数個要求に対する割当てを説明する。なお、Ｓ２１～Ｓ３１は各処理ステップを示す。

【００４３】ステップＳ２１において、利用可能なＰＥを検索する。ステップＳ２２において、要求される個数の利用可能なＰＥの全ての組み合わせに対して、第１ループを構成する。すなわち、要求される個数の利用可能なＰＥの全ての組み合わせに対して、以下のステップを繰り返す。

【００４４】ステップＳ２３において、当該プロセスの割当てによって生じる通信リンク２０のデータ転送量を計算する。ステップＳ２４において、通信リンク２０のデータ転送量の最大値が最小の割当てを保持する。ステップＳ２５において、第１のループを終了する。ステップＳ２６において、通信リンク２０のデータ転送量の増加が最小であるような割当てを第１のループで選択された割当てとして、全ての選択された割当てに対して第２のループを構成する。

【００４５】ステップＳ２７において、選択された割当て方で、複数個のＰＥを仮割当てする。ステップＳ２８において、次の割当て効率を計算する。なお、次の割当て効率を計算した結果を、Ｒｅｓｕｌｔと記す。ステップＳ２９において、結果Ｒｅｓｕｌｔが最小の割当てを保持する。ステップＳ３０において、第２のループを終了する。ステップＳ３１において、Ｒｅｓｕｌｔが最小の割当てである割当て方で、実際にプロセスに割当てる。以上の処理を終了する。

【００４６】(4)：次の割当て効率計算処理の説明・・・図１３参照

図１３は従来例の説明図（その６）であり、図１１のステップＳ１４及び図１２のステップＳ２８に於ける次の割当て効率計算処理を示すフローチャートである。以下、図１３に基づいて、前述した次の割当て効率計算処理を説明する。なお、Ｓ４１～Ｓ４３は各処理ステップを示す。

【００４７】ステップＳ４１において、利用可能なＰＥの内で、最も左端のＰＥを選択してＰＥ－Ｌとする。ステップＳ４２において、利用可能なＰＥの内でも最も右端のＰＥを選択してＰＥ－Ｒとする。ステップＳ４３において、ＰＥ－ＬからＰＥ－Ｒまでの通信リンク数を求め、この通信リンク数をＲｅｓｕｌｔとする。以上で処理を終了する。

【００４８】図１３のフローチャートにおいては、最も左端のＰＥと最も右端のＰＥを選択して、両ＰＥ間での通信リンク数を求める。すなわち、この通信リンク数によって、次のプロセスを割当てる際の割当て効率を求めていることになる。これは以下のように考えれば良い。すなわち、両ＰＥ間での通信リンク数が少ないということは、利用可能なＰＥがまとまって存在していることを意味する。逆に、両ＰＥ間での通信リンク数が多いと、利用可能なＰＥが広い範囲に広がって存在していることになる。

【００４９】狭い範囲にまとまったＰＥにプロセスを割当てる方が、当然ながら割当てられたＰＥ間に介在するＰＥの数も少なく、割当て後のデータ転送量の最大値も小さくなる可能性が高い。広い範囲で互いに離れたＰＥにプロセスを割当てたのでは介在するＰＥの数が多く、別のプロセスのデータ転送と重なる可能性が高く、従っ

て、割当て後のデータ転送量の最大値も大きくなる可能性が高い。

【００５０】すなわち、図１３のフローチャートは、プロセス割当て後に残された利用可能なＰＥがまとまってなるべく狭い範囲に存在するような指標を与えることになる。すなわち、プロセス割当て後に、残された利用可能なＰＥを次回割当てる際に、データ転送効率がなるべくよくなるような指標を与えていることになる。

【００５１】

【発明が解決しようとする課題】前記のような従来のものにおいては、次のような課題があった。

(1)：従来例１では、プロセッサの能力はまだ十分に開いているにも関わらず、データ転送ができないため、全体の処理が実時間で実現不可能になってしまうことがある。

【００５２】(2)：従来例２では、前記プロセス割当て方法は通信経路が複数あれば効果はあるが、リニアアレイ型プロセッサシステムでは、通信路は１つ、リンク構成でも２つしかなく、ほとんど効果がない。

【００５３】(3)：従来例３では、制御グループを先に静的に決定してしまうため、グループ間のデータ転送量の負荷分散を行うには、その都度、制御グループの割当て変更が必要となり、負荷分散が動的に行われない。

【００５４】(4)：従来例４に示したリニアアレイ型プロセッサシステムにおける動的プロセス割当てにおいて、割当てるべきプロセスは複数のプロセッサ資源を要求する。従って、一般的に、各プロセスは単一プロセッサで処理されるサブプロセスを複数接続した構成となっている。サブプロセスの実行スケジュールによっては、そのプロセスが要求するプロセッサ数や、データ転送量は可変である。

【００５５】従って、このサブプロセスの実行スケジュールが動的プロセス割当てによるプロセッサ間データ転送量の増減に影響し、このことで、システム全体の能力を充分に発揮することができなくなってしまうという課題がある。

【００５６】また、プロセスが要求するプロセッサ数やデータ転送量の少ないサブプロセスの実行スケジュールを使うことで、他のプロセスの実行速度は満足できても、そのプロセス単体の処理速度が低下してしまい、全体の処理が実現できなくなるという課題がある。

【００５７】本発明は、このような従来の課題を解決し、リニアアレイ型プロセッサシステムにおける動的プロセス割当てによってシステム全体の能力を最大限引き出すために、サブプロセスの実行スケジュールを、予め静的スケジューリング手段によって求めておくことで、効率的な処理を可能にすることを目的とする。

【００５８】

【課題を解決するための手段】本発明は前記の目的を達成するため、次のように構成した。

(1)：複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるプロセス割当て方法において、前記動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、サブプロセスの静的スケジューリング手段により静的に求めておくようにした。

【００５９】(2)：前記(1)のプロセス割当て方法において、前記サブプロセスの静的スケジューリング手段が、そのプロセスが要求するプロセッサ数最小となるスケジュール情報を得るようにした。

【００６０】(3)：前記(1)のプロセス割当て方法において、前記サブプロセスの静的スケジューリング手段が、そのプロセス単体の処理時間最小となるスケジュール情報を得るようにした。

【００６１】(4)：前記(1)のプロセス割当て方法において、前記サブプロセスの静的スケジューリング手段が、そのプロセスが要求するプロセッサ間データ転送量最小となるスケジュール情報を得るようにした。

【００６２】(5)：前記(1)のプロセス割当て方法において、前記サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュール情報を選択して割当てるようにした。

【００６３】(6)：複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるためのサブプロセスのスケジュール情報を、予め、静的に求めておくサブプロセスの静的スケジューリング手段を備えていることを特徴とした静的スケジューリング装置。

【００６４】(7)：コンピュータに、複数のプロセッサを直列接続したリニアアレイ型マルチプロセッサシステムの前記各プロセッサに対し、複数のサブプロセスを構成要素として含むプロセスを動的に割当てるためのサブプロセスのスケジュール情報を、予め、静的に求めておく手順を実行させるためのプログラムを記録したコンピュータ読取可能な記録媒体。

【００６５】（作用）前記構成に基づく本発明の作用を説明する。

(a)：前記(1)の作用

サブプロセスの静的スケジューリング手段は、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセスを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【００６６】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄

与するところが大である。

【００６７】(b) ：前記(2) の作用

サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセスが要求するプロセッサ数最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセスが要求するプロセッサ数が最小となるサブプロセススケジュールを求めることで、プロセッサ数の少ないシステムに適した動的プロセス割当てを行うことができる。

【００６８】(c) ：前記(3) の作用

サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセス単体の処理時間最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセス単体の処理時間が最小となるサブプロセススケジュールを求めることで、全プロセスの処理時間の短い動的プロセス割当てを行うことができる。

【００６９】(d) ：前記(4) の作用

サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセスが要求するプロセッサ間データ転送量最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセスが要求するプロセッサ間データ転送量の最大値が最小となるサブプロセススケジュールを求めることで、プロセッサ間データ転送のための、容量が小さなシステムに適した動的プロセス割当てを行うことができる。

【００７０】(e) ：前記(5) の作用

サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、前記サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュール情報を選択して割当てる。

【００７１】このようにすれば、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、複数のサブプロセススケジュールを求めておき、その中から動的プロセス割当て時に、最適なサブプロセススケジュールを選択することで、その時の割当て状況に適応した動的プロセス割当てを行うことができる。

【００７２】(f) ：前記(6) の作用

静的スケジューリング装置のサブプロセスの静的スケジューリング手段は、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセスを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【００７３】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄与するところが大である。

【００７４】(g) ：前記(7) の作用

コンピュータが、前記記録媒体のプログラムを読み出して実行することにより、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセスを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【００７５】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄与するところが大である。

【００７６】

【発明の実施の形態】以下、発明の実施の形態を図面に基づいて詳細に説明する。

§１：システムの説明・・・図１参照

図１はシステムの説明図である。以下、図１に基づいて本実施の形態で使用するシステムについて説明する。図１に示したように、このシステムは、リニアアレイ型マルチプロセッサシステムに対し、動的プロセス割当てを行うシステムである。

【００７７】前記リニアアレイ型マルチプロセッサシステムは、複数のプロセッサ３２（ＰＥ１、ＰＥ２、ＰＥ３、ＰＥ４・・・）が直列接続されたシステムであり、前記従来例４と同じ構成のシステムである。この場合、前記プロセッサ３２は、それぞれ、前記従来例４の情報処理ユニット１０に対応している。

【００７８】そして、各プロセッサ３２に対して、動的プロセス割当てを行うホストプロセッサ３１を前記各プロセッサ３２とバスにより接続されている。また、このホストプロセッサ３１により、前記各プロセッサ３２に動的プロセス割当てを行う前に、サブプロセスの静的割当てを行うための静的スケジューリング装置３３を前記ホストプロセッサ３１に接続しておく。

【００７９】ホストプロセッサ３１には、各プロセッサ３２に対して動的プロセス割当てを行うための動的プロセス割当て手段３４が設けてあり、静的スケジューリング装置３３には、サブプロセスの静的スケジューリングを行うためのサブプロセスの静的スケジューリング手段３５が設けてある。

【００８０】この場合、静的スケジューリング装置３３は、リニアアレイ型プロセッサシステムに対し、バス結合等により直接接続しても良いが、このような接続でなくても良い。例えば、静的スケジューリング装置３３

と、ホストプロセッサ３１の間は、何らかの手段によりデータ通信が可能な状態にしておけば良く、ＬＡＮ等の通信回線を介して接続し、互いにデータ伝送可能な状態で使用することも可能である。

【００８１】すなわち、プロセッサ３３は、前記ホストプロセッサ３１の近く（例えば、同一パーソナルコンピュータ内）に置いても良いし、離れた場所（異なる装置内）に置いても良い。少なくとも、動的プロセス割当て手段３４を備えた装置へデータ伝送できれば実施可能である。この場合、前記動的プロセス割当て手段３４、及びサブプロセスの静的スケジューリング手段３５は、プログラムの実行により実現するものである。

【００８２】§２：プロセス割当て方法の説明・・・図２参照

図２はサブプロセスの処理説明図である。以下、図２に基づいて、サブプロセスの処理を説明する。図１に示したシステムにおいて、ホストプロセッサ３１が各プロセッサ３２に対し、動的プロセス割当て処理を行うことで、前記リニアアレイ型プロセッサシステムによる処理を行う。

【００８３】この場合、割当てる各プロセスは、一般に複数のプロセッサ資源を要求するため、単一プロセッサで処理されるサブプロセスを複数接続した構成になっている。このサブプロセスの実行によって、そのプロセスが要求する資源は変化する。以下の説明では、前記サブプロセスは、１つのプロセスの要素を構成しており、１つのプロセスに複数のサブプロセスが存在する場合について説明する。

【００８４】前記のように、ホストプロセッサ３１が動的プロセス割当てを行う時、前記動的割当てを行うプロセスを構成する複数のサブプロセスを、動的プロセス割当てを行う前に、静的スケジューリング装置３３に設けたサブプロセスの静的スケジューリング手段３５が、静的にスケジューリングを行う。

【００８５】そこで求められたサブプロセスの実行スケジュール情報から、そのプロセスの要求するプロセッサ数や、プロセッサ間データ転送量を、前記ホストプロセッサ３１へ転送し、動的プロセス割当て手段３４が受信する。その後、ホストプロセッサ３１の動的プロセス割当て手段３４は、受信した前記プロセスの要求するプロセッサ数や、プロセッサ間データ転送量を基に、各プロセッサ３２に対して動的プロセス割当てを行う。このようにして、全体の処理の効率の良い実現を可能にしている。

【００８６】この場合、前記リニアアレイ型マルチプロセッサシステムの各プロセッサ３２に対し、サブプロセスを構成要素として含むプロセスの割当てを行うプロセス割当て方法には、次のような方法がある。

【００８７】①：プロセス割当て方法１は、前記サブプロセスの静的スケジューリング手段３５が、そのプロセスが要求するプロセッサ数最小となるスケジュールを得る方法（時間制約付スケジューリング）である。

【００８８】②：プロセス割当て方法２は、サブプロセスの静的スケジューリング手段３５が、そのプロセス単体の処理時間が最小となるスケジュールを得る方法（プロセッサ数制約付スケジューリング）である。

【００８９】③：プロセス割当て方法３は、サブプロセスの静的スケジューリング手段３５が、そのプロセスが要求するプロセッサ間データ転送量が最小となるスケジュールを得る方法である。

【００９０】④：プロセス割当て方法４は、前記サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュールを選択して割当てる方法である。

【００９１】§３：プロセス割当て方法１の説明・・・図３参照

図３はプロセス割当て処理フローチャート（その１）である。プロセス割当て方法１は、サブプロセスの静的スケジューリング手段３５が、そのプロセスが要求するプロセッサ数が最小となるスケジュールを得る方法（時間制約付スケジューリング）であり、以下、図３に基づいて説明する。なお、以下の処理はサブプロセスの静的スケジューリング手段３５が行う処理であり、Ｓ５１～Ｓ５４は各処理ステップを示す。

【００９２】サブプロセスの静的スケジューリング手段３５は静的スケジューリングを行う時、指定した時間内にそのプロセスが終了するために要求されるプロセッサ数Ｎｐが最小となるサブプロセススケジュールを求める。従って、先ず、前記プロセッサ数Ｎｐを、Ｎｐ＝（サブプロセスの処理時間の総和）／（目的の処理時間）の式により求める（Ｓ５１）。

【００９３】次に、Ｎｐプロセッサで割当てを試み（Ｓ５２）、割当てが可能か否かを判断する（Ｓ５３）。その結果、割当てが可能であれば処理を終了するが、割当てが不可能であれば、Ｎｐを更新（Ｎｐ＝Ｎｐ＋１）し（Ｓ５４）、前記Ｓ５２の処理から繰り返して行う。このようにして、前記Ｓ５３の処理で割当てが可能になった時点で、そのプロセスが要求するプロセッサ数が最小となるスケジュールを得ることができる。

【００９４】§４：プロセス割当て方法２の説明・・・図４参照

図４はプロセス割当て処理フローチャート（その２）である。プロセス割当て方法２は、サブプロセスの静的スケジューリング手段３５が、そのプロセス単体の処理時間が最小となるスケジュールを得る方法（プロセッサ数制約付スケジューリング）であり、以下、図４に基づいて説明する。なお、以下の処理はサブプロセスの静的スケジューリング手段３５が行う処理であり、Ｓ６１～Ｓ６４は各処理ステップを示す。

【００９５】サブプロセスの静的スケジューリング手段

３５は静的スケジューリングを行う時、指定したプロセッサ数でそのプロセスが終了するための処理時間Ｔが最小となるサブプロセススケジュールを求める。従って、先ず、前記処理時間Ｔを、Ｔ＝（サブプロセスの処理時間の総和）／（目的のプロセッサ数）の式により求める（Ｓ６１）。

【００９６】次に、処理時間≦Ｔの条件で割当てを試み（Ｓ６２）、割当てが可能か否かを判断する（Ｓ６３）。その結果、割当てが可能であれば処理を終了するが、割当てが不可能であれば、Ｔを更新｛Ｔ＝Ｔ＋（単位時間）｝し（Ｓ６４）、前記Ｓ６２の処理から繰り返して行う。このようにして、前記Ｓ６３の処理で割当てが可能になった時点で、そのプロセス単体の処理時間が最小となるスケジュールを得ることができる。

【００９７】§５：プロセス割当て方法３の説明・・・図５参照

図５はプロセス割当て処理フローチャート（その３）である。また、図６はプロセス割当て処理説明図（その１）である。プロセス割当て方法３は、サブプロセスの静的スケジューリング手段３５が、そのプロセスが要求するプロセッサ間データ転送量が最小となるスケジュールを得る方法であり、以下、図５、図６に基づいて説明する。なお、以下の処理はサブプロセスの静的スケジューリング手段３５が行う処理であり、Ｓ７１～Ｓ７９は各処理ステップを示す。また、Ｎはサブプロセスの数、Ｍｉはｉ分割の分割パターン数である。

【００９８】サブプロセスの静的スケジューリング手段３５は、先ず、プロセスを、サブプロセスの数をＮとした時、ｉ＝１からＮまで分割を試みる。すなわち、ｉ＝１～Ｎループとして（Ｓ７１）、プロセスを分割する（Ｓ７２）。この場合、ｉ分割する時の分割パターンとの数をＭｉとすると、分割パターンは、ｊ＝１からＭｉループまで存在する（Ｓ７３）。

【００９９】例えば、Ｎ＝３で、｛1，2，3｝を２分割する場合、｛｛1，2｝，｛3｝｝，｛｛1｝，｛2，3｝｝のＭ２＝２パターン存在する。従って、ｊ＝１～Ｍｉループとして（Ｓ７３）、分割されたプロセスに対してスケジューリングを行い、各サブプロセスの実行開始時刻を求める（Ｓ７４）。この時、条件（時間制約）を満たすことができるか否かを判断し（Ｓ７５）、前記条件を満たせばＳ７６へ進み、満たさなければ、Ｓ７８へ進み、別の分割が試される。

【０１００】前記条件を満たした場合（時間制約を満たす場合）、その時の各分割間のデータ転送量の最大値が最小となるか否かを判断し（Ｓ７６）、最小とならなければＳ７８の処理を行い、最小となれば、データ転送量の最大値が最小となるものをメモリ等に保存する（Ｓ７７）。

【０１０１】これを繰り返し実行し、全ての分割パターンに対して行う。そして、２重ループ（Ｓ７８、Ｓ７９）が終了した時、前記メモリ等に保存されているスケジュールをそのプロセスのサブプロセススケジュールとする。

【０１０２】以下、図６に基づいて、前記プロセス割当て方法３の処理を具体的に説明する。図６に示したプロセスは、８個のサブプロセス｛1，2，3，4，5，6，7，8｝から構成される。これをｉ＝３の時、すなわち、３分割の場合、例えば、｛1，5，7｝，｛2，4｝，｛3，6，8｝のように分割される。

【０１０３】この結果に対してスケジューリングを行い、３プロセッサに対して、各サブプロセスの開始時刻を決定する。そして、条件（時間制約）を満たすスケジュールが存在すれば、転送量を計算し、その内の最大値が、以前に求めた割当てより小さい場合は、これを保存する。

【０１０４】§６：プロセス割当て方法４の説明・・・図７参照

図７はプロセス割当て処理説明図（その２）である。プロセス割当て方法４は、サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュールを選択して割当てる方法であり、以下、図７に基づいて説明する。

【０１０５】この処理では、前記サブプロセスの静的スケジューリング手段３５がＮ個で構成されており、これらのＮ個の手段によりサブプロセスの静的スケジューリングを行う。前記静的スケジューリングを行う時、前記Ｎ個のサブプロセスの静的スケジューリング手段（１～Ｎ）により、予め求めておいた複数のサブプロセススケジュールから、動的プロセス割当て時に最適なサブプロセススケジュール結果を選択して使用する。

【０１０６】１．現在のプロセッサへのプロセス割当て状況（どのプロセッサが空いているか）
２．現在のプロセッサ間のデータ転送量
３．将来のプロセス割当て解除情報（いつ、どのプロセッサから、どのプロセスが割当て解除されるか）
前記処理において、前記１～３に示した３つの情報をプロセスに割当てる毎に更新している。次のプロセスを割当てる時、この３つの情報から例えば、プロセッサの空きが少ない場合は、要求プロセッサ数が最小となるサブプロセッサ割当てを選択してプロセス割当てを行う。また、データ転送量の空きが少ない時には、データ転送量が最小となるサブプロセッサ割当てを選択してプロセス割当てを行う。

【０１０７】更に、次に行うプロセスの処理時間が最小となるサブプロセス割当ては、この処理時間と割当て解除になる時間との和が、このプロセスの時間制約以内に納まる時使用される。

【０１０８】§７：記録媒体とプログラムの説明

前記静的スケジューリング装置３３のサブプロセスの静的スケジューリング手段３５が行うサブプロセスの静的

スケジューリング処理は、静的スケジューリング装置３３内のＣＰＵがプログラムを実行することにより、次のようにして実現する。

【０１０９】静的スケジューリング装置３３にはハードディスク装置が設けてあり、このハードディスク装置の記録媒体（ハードディスク）に、前記処理を実現するためのプログラムやその他の各種データ等を格納しておく。そして、前記処理を行う場合は、ＣＰＵの制御によりハードディスク装置の記録媒体に格納されている前記プログラムやデータを読み出して静的スケジューリング装置３３内に設けたメモリに取り込む。

【０１１０】その後、ＣＰＵが前記メモリに格納してあるプログラムの内、必要なプログラムから順次読み出して実行することにより、前記処理を行う。なお、前記ハードディスク装置の記録媒体に記録するプログラムは、次のようにして記録（記憶）する。

【０１１１】①：フレキシブルディスク（フロッピィディスク）に格納されているプログラム（他の装置で作成したプログラムデータ）を、静的スケジューリング装置３３に設けたフレキシブルディスクドライブ装置により読み取り、ハードディスク装置の記録媒体（ハードディスク）に格納する。

【０１１２】②：光磁気ディスク、或いはＣＤ－ＲＯＭ等の記憶媒体に格納されているデータを、静的スケジューリング装置３３に設けたドライブ装置により読み取り、ハードディスク装置の記録媒体（ハードディスク）に格納する。

【０１１３】③：ＬＡＮ等の通信回線を介して他の装置から伝送されたデータを静的スケジューリング装置３３で受信し、そのデータをハードディスク装置の記録媒体（ハードディスク）に格納する。

【０１１４】

【発明の効果】以上説明したように、本発明によれば次のような効果がある。(1)：請求項１では、サブプロセスの静的スケジューリング手段は、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセスを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【０１１５】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄与するところが大である。

【０１１６】(2)：請求項２では、サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセスが要求するプロセッサ数最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセスが要求するプロセッサ数が最小となるサブプロセススケジュールを求めることで、プロセッサ数の少ないシステムに適した動的プロセス割当てを行うことができる。

【０１１７】(3)：請求項３では、サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセス単体の処理時間最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセス単体の処理時間が最小となるサブプロセススケジュールを求めることで、全プロセスの処理時間の短い動的プロセス割当てを行うことができる。

【０１１８】(4)：請求項４では、サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、そのプロセスが要求するプロセッサ間データ転送量最小となるスケジュール情報を得る。このように、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、そのプロセスが要求するプロセッサ間データ転送量の最大値が最小となるサブプロセススケジュールを求めることで、プロセッサ間データ転送のための、容量が小さなシステムに適した動的プロセス割当てを行うことができる。

【０１１９】(5)：請求項５では、サブプロセスの静的スケジューリング手段は、サブプロセスのスケジュール情報を求める時、前記サブプロセスのスケジュール情報を、複数の条件により求めておき、プロセスの要求条件に応じて適当なスケジュール情報を選択して割当てる。

【０１２０】このようにすれば、サブプロセスの静的スケジューリング手段が静的スケジューリングを行う時、複数のサブプロセススケジュールを求めておき、その中から動的プロセス割当て時に、最適なサブプロセススケジュールを選択することで、その時の割当て状況に適応した動的プロセス割当てを行うことができる。

【０１２１】(6)：請求項６では、静的スケジューリング装置のサブプロセスの静的スケジューリング手段は、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセスを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【０１２２】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄与するところが大である。

【０１２３】(7)：請求項７では、コンピュータが、前記記録媒体のプログラムを読み出して実行することにより、動的プロセス割当てを行うためのサブプロセスのスケジュール情報を、予め、静的に求めておく。このように、動的プロセス割当てを行う時、割当てを行うプロセ

19

スを構成するサブプロセスを動的プロセス割当てを行う前に、予め静的にスケジュール情報を求めておく。

【０１２４】そして、そこで求められたサブプロセスの実行スケジュールからそのプロセスの要求するプロセッサ数やプロセッサ間データ転送量を基に、動的プロセス割当てを行うことで、全体の処理の効率の良い実現に寄与するところが大である。

【図面の簡単な説明】

【図１】本発明の実施の形態におけるシステムの説明図である。

【図２】本発明の実施の形態におけるサブプロセスの処理説明図である。

【図３】本発明の実施の形態におけるプロセス割当て処理フローチャート（その１）である。

【図４】本発明の実施の形態におけるプロセス割当て処理フローチャート（その２）である。

【図５】本発明の実施の形態におけるプロセス割当て処理フローチャート（その３）である。

【図６】本発明の実施の形態におけるプロセス割当て処理説明図（その１）である。

【図７】本発明の実施の形態におけるプロセス割当て処理説明図（その２）である。

【図８】従来例の説明図（その１）である。

20

*【図９】従来例の説明図（その２）である。

【図１０】従来例の説明図（その３）である。

【図１１】従来例の説明図（その４）である。

【図１２】従来例の説明図（その５）である。

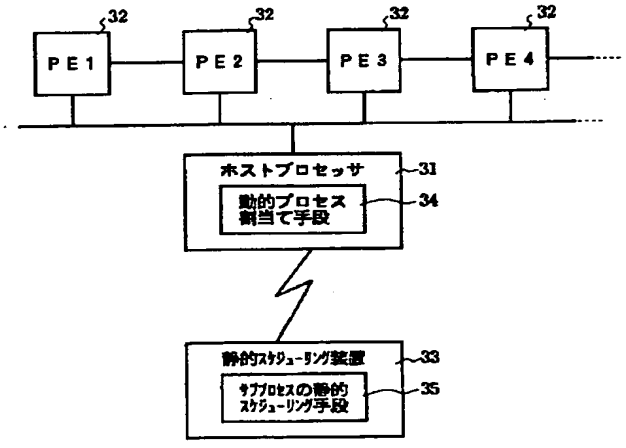
【図１３】従来例の説明図（その６）である。

【符号の説明】

１０　情報処理ユニット
１１　信号処理プロセッサ
１２　命令キャッシュ
１３　データＲＡＭ
１４、１５　リンク制御部
１６　メインキャッシュ
１７　リンクキャッシュ
１８　ＤＲＡＭ
１９　ＤＲＡＭコントローラ
２０　通信リンク
２５　信号処理部
２６　通信制御部
３０　ホストメモリバス
３１　ホストプロセッサ
３３　静的スケジューリング装置
３４　動的プロセス割当て手段
３５　サブプロセスの静的スケジューリング手段

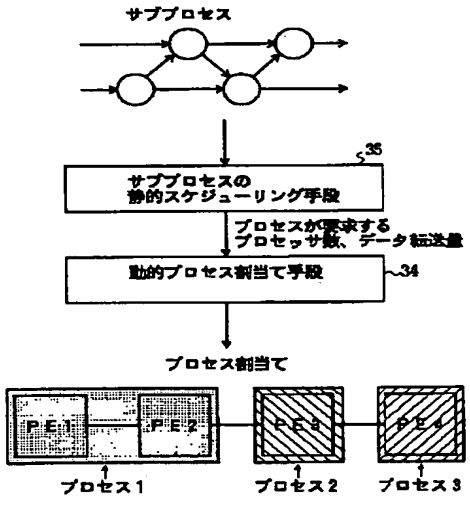
【図１】

システムの説明図



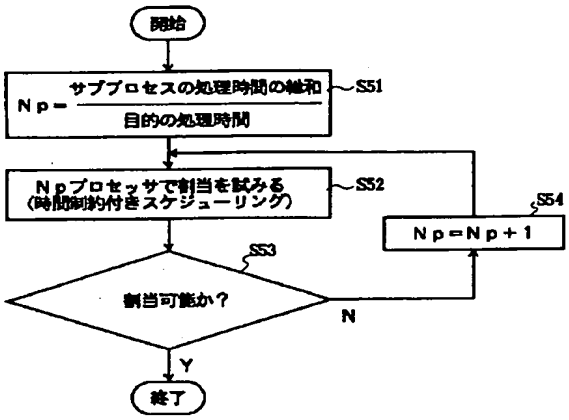
【図２】

サブプロセスの処理説明図



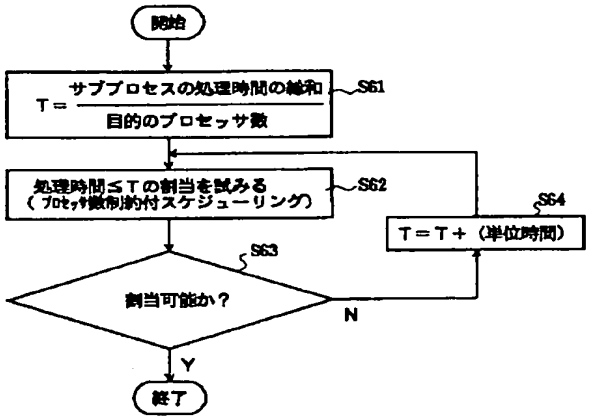
【図３】

プロセス割当て処理フローチャート（その１）



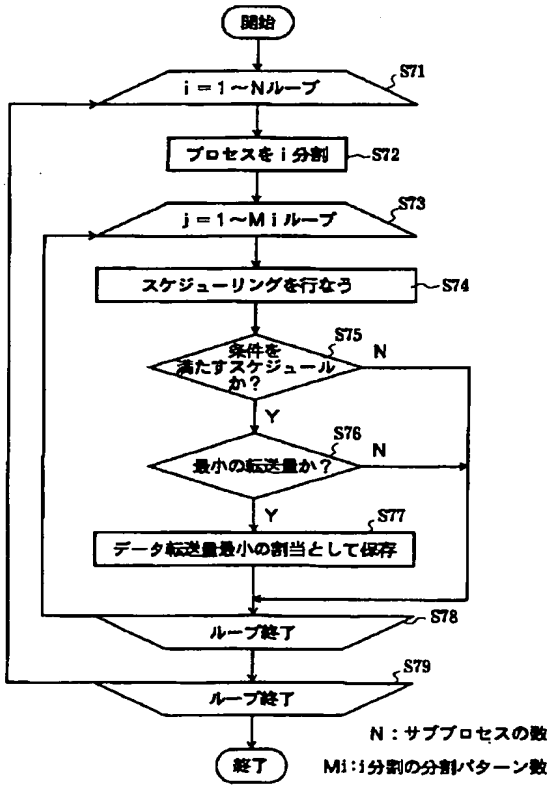
【図４】

プロセス割当て処理フローチャート（その２）



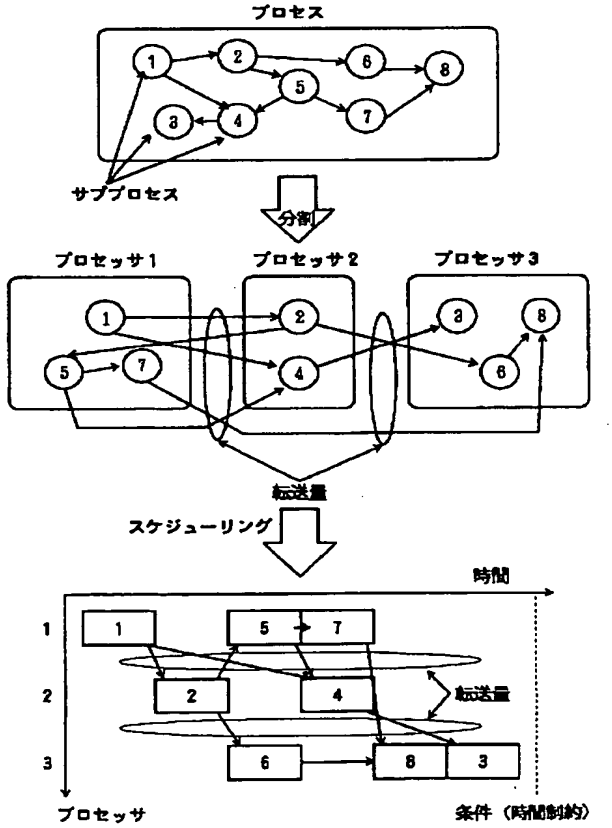
【図５】

プロセス割当て処理フローチャート（その３）



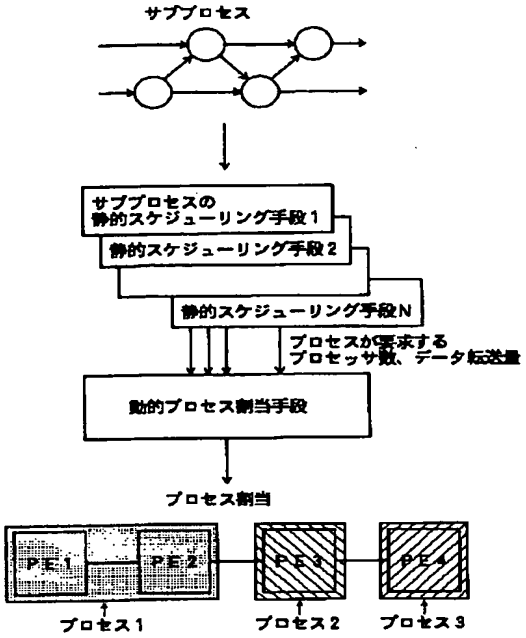
【図６】

プロセス割当て処理説明図（その１）



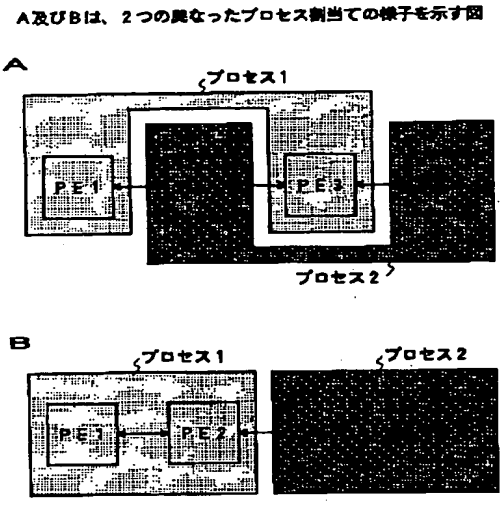
【図７】

プロセス割当て処理説明図（その２）



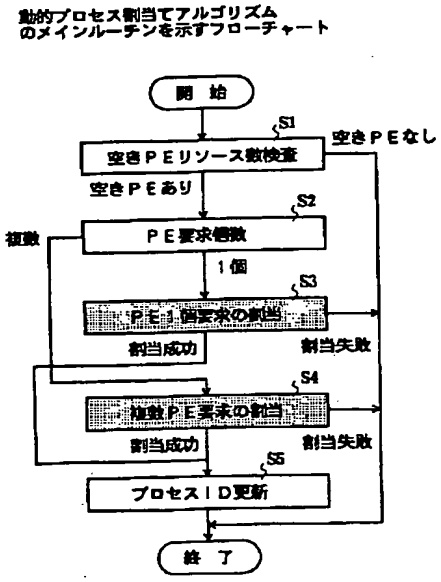
【図９】

従来例の説明図（その２）



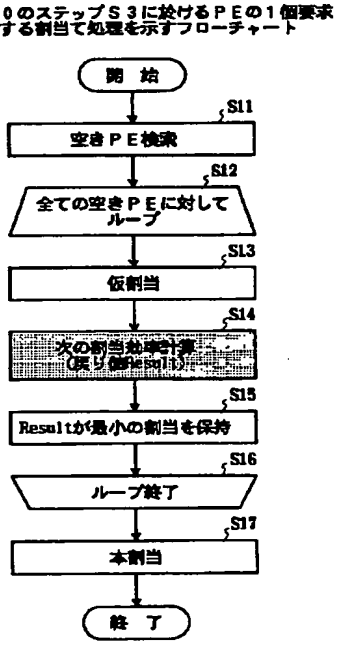
【図１０】

従来例の説明図（その３）



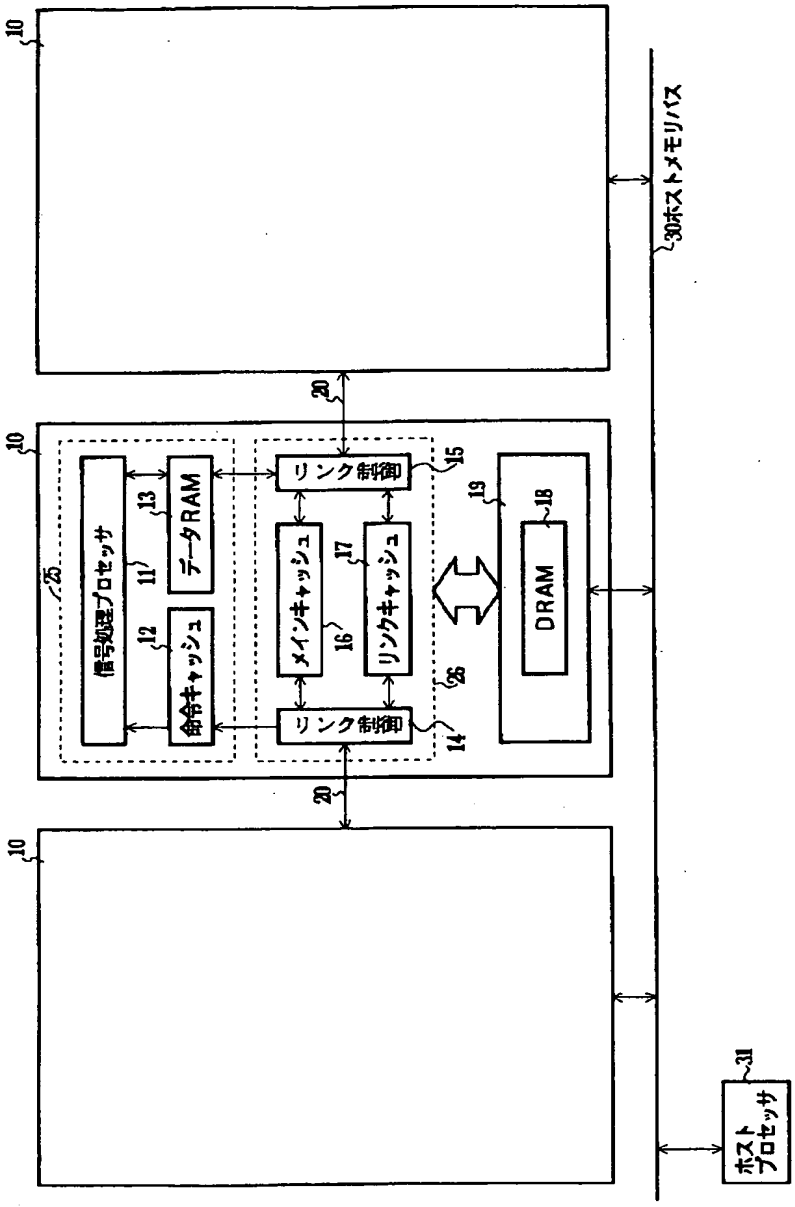
【図１１】

従来例の説明図（その４）



【図８】

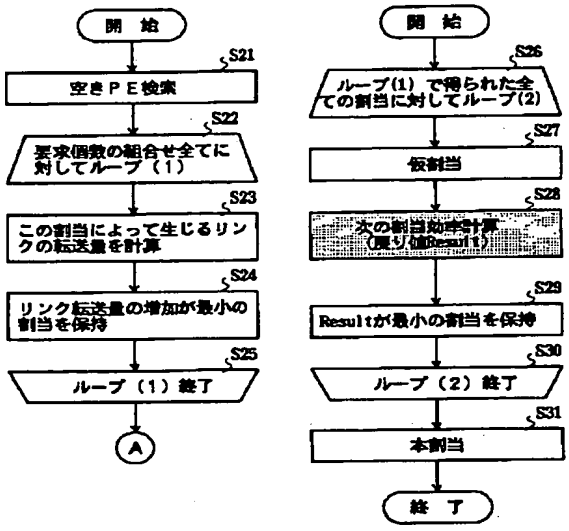
従来例の説明図（その１）



【図１２】

従来例の説明図（その５）

図１０のステップＳ４に於けるＰＥの複数個要求に対する割当てを示すフローチャート



【図１３】

従来例の説明図（その６）

図１１のステップ１４及び図１２のステップ２８に於ける次の割当て効率計算処理を示すフローチャート